Hashing IV and Course Overview

Other Collision Resolution Techniques

1) Double Hashing

- The first hash function determines the home address.
- If the home address is occupied, apply a second hash function to get a number c (c relatively prime to N).
- c is added to the home address to produce an overflow addresses; if occupied, proceed by adding c to the overflow address, until an empty spot is found.
- This is an atempt to reduce clustering.

Example:

k (key)	ADAMS	JONES	MORRIS	SMITH
$h_1(k)$ (home address)	5	6	6	5
$h_2(k) = c$	2	3	4	3

Hashed file using double hashing:

0	
1	
2	
3	
4	
5	ADAMS
6	JONES
7	
8	SMITH
9	
10	MORRIS

Suppose the above table was full, and that a key $\overline{k}$ had $h_1(\overline{k}) = 6$ and $h_2(\overline{k}) = 3$.

Question: What would be the order in which the addresses would be probed when trying to insert $\overline{k}$?

Answer: 6, 9, 1, 4, 7, 10, 2, 5, 8, 0, 3.

2) Chained Progressive Overflow

- Similar to progressive overflow, except that synonyms are linked together with pointers.
- The objective is to reduce the search length for records within clusters.

Example X:

Key	**Home Address**	**Search Length with Progressive Overflow**	**Search Length with Chained Progressive Overflow**
ADAMS	20	1	1
BATES	21	1	1
COLES	20	3	2
DEAN	21	3	2
EVANS	24	1	1
FLINT	20	6	3
Average Search Length :		2.5	1.7

Progressive Overflow

	data
⋮	⋮
20	ADAMS
21	BATES
22	COLES
23	DEAN
24	EVANS
25	FLINT
⋮	⋮

Chained Progressive Overflow

	data	**next**
⋮	⋮	⋮
20	ADAMS	22
21	BATES	23
22	COLES	25
23	DEAN	-1
24	EVANS	-1
25	FLINT	-1
⋮	⋮	⋮

PROBLEM: Suppose that 'DEAN' home address is 22. Since 'COLES' is there, we couldn't have a link to 'DEAN' starting in its home address!

Solution:
Two-pass loading:
- First pass: only load records that fit into their home addresses.
- Second pass: load all overflow records.
Care should be taken when deletions are done.

key	**home address**
ADAMS	20
BATES	21
COLES	20
DEAN	22
EVANS	24
FLINT	20

table after first pass:

20	ADAMS	-1
21	BATES	-1
22	DEAN	-1
23		
24	EVANS	-1
25		

table after second pass:

20	ADAMS	23
21	BATES	-1
22	DEAN	-1
23	COLES	25
24	EVANS	-1
25	FLINT	-1

3) Chaining with a Separate Overflow Area

Move overflow records to a **Separate Overflow Area**.
A linked list of synonyms start at their home address in the Primary data area, continuing in the separate overflow area.

Example X, with separate overflow area:

primary data area

20	ADAMS	0
21	BATES	1
22		
23		
24	EVANS	-1
25		

overflow area

0	COLES	2
1	DEAN	-1
2	FLINT	-1
3		
	⋮	⋮

When the packing density is higher than 1 an overflow area is **required**.

4) Scatter Tables: Indexing Revisited

Similar to chaining with separate overflow, but the hashed file contains no records, but only pointers to data records. The scatter

Example X organized as scatter table:

index (hashed)

	⋮
20	0
21	1
22	
23	
24	4
	⋮

datafile (entry-sequenced, sorted, etc.)

	data	next
0	ADAMS	2
1	BATES	3
2	COLES	5
3	DEAN	-1
4	EVANS	-1
5	FLINT	-1

Note that the data file can be organized in many different ways: sorted file, entry sequenced file, etc.

Patterns of Record Access

> Twenty percent of the students send 80 percent of the e-mails.
> *L. M.*

Using knowledge of pattern of record access to improve performance ...

Suppose you know that 80% of the searches occur in 20% of the items.

How to use this info to try to reduce search length in a hashed file ?

- Keep track of record access for a period of time (say 1 month).
- Sort the file in descending order of access.
- Re-hash using this order.

Records more frequently searched are more likely to be **at** or **close to** their home addresses.

Course Overview

Fundamental file processing operations

- open, close, read, write, seek (file as a stream of bytes)

Secondary Storage Devices and System Software

- how different secondary storage devices work (tapes, magnetic disks, CD-ROM)
- the role of different basic software and hardware in I/O
(operating system (file manager), I/O processor, disk controller)
- buffering at the level of the system I/O buffers.

Logical view of files and file organization

- file as a collection of records (concepts of records, fields, keys)
- record and field structures
- sequential access; direct access (RRN, byte offset).

Organizing files for performance

- data compression
- reclaiming space in files: handling deletions, AVAIL LIST, etc.
- sorting and searching: internal sorting, binary searching, keysorting.

Cosequential processing

- main characteristics: **sequential access** to input and output files, and **co-ordinated access** of input files.
- main types of processing: matching (intersection) and merging (union).
- main types of application: the merging step in an external sorting method; posting of transactions to a master file.

Indexing

- Primary and secondary key indexes.
- Maintenance of indexed files; different index organizations.
- Alternative ways of organizing an index:
  - Simple index
    keeping index sorted by key (additions and deletions are expensive: about $O(n)$ disk accesses)
  - B trees and B+ trees
    improvement in time: searches, insertions and deletions in about $O(\log_k n)$ disk accesses, where k is the order of the tree and n is the number of records for B trees or the number of blocks of records for B+ trees.
  - Hashed index: searches, insertions and deletions in constant expected time, i.e. expected time $O(1)$.

B trees and B+ trees

- We started from the problem of maintenance of simple indexes.
- B trees: multi-level index that work from bottom up and guarantees searches and updates in about $O(\log_k n)$ disk accesses.
- Then, we discussed the problem of having both an indexed and a sequential view of the file: B+ trees.
- B+ tree = sequence set + index set
  The index set is a B tree; the sequence set is like a doubly linked list of **blocks** of records.
- simple prefix B+ trees: B+ trees in which we store separators (short prefixes) rather than keys, in the index set.
- We learned B trees and B+ trees maintenance and operations.

Hashing

- We have discussed static hashing techniques: expected time for access is $O(1)$ when the file doesn't change too much.

- We have discussed: hash functions, record distribution and search length, the use of buckets, collision resolution techniques (progressive overflow, chained progressive overflow, chaining with a separate overflow area, scatter tables, patterns of record access).

- We have not discussed, but it's worth mentioning, **extendible hashing**. In extendible hashing, hashing is modified to become self-adjusting, allowing for the desired expected $O(1)$ access even when the file grows a lot; the address space changes after a lot of insertions or deletions. If you are curious, read Chapter 12.

What's next: Database Management Systems CSI3317

Different layers:

Database management systems
File systems
Physical storage devices

Each layer hides details of the lower level.
This course focused in **file processing**. In order to do file processing efficiently we studied some key issues concerning **physical storage devices**.
CSI3317 will focus on **database management systems**.

As the number of users and applications in an organization grows, file processing evolves into database processing.
A **database management system** is a large software that maintains the data of an organization and mediates between the data and the application programs.
Each application program asks the DBMS for data, the DBMS figures out the best way to locate the data.
The applications are coded without knowing the physical organization of the data.
File processing is done by the DBMS rather than the application program.
The application program describes the database at a high-level, conceptual view; this high-level description is called a data model. One of the most popular data models is the relational data model; SQL is a commercially used, standard relational-database language.

Difficulties and issues handled by a DBMS:

- data independence: changes in file organization should not require changes in the application programs; example of common changes: add new fields to records of a file, add an index to a file, add and remove secondary indexes.
- data sharing: care must be taken when several users may be reading and modifying the data.
- data integrity: ensuring consistency of data (when data is updated, related data must be updated)

You you learn a lot about this in CSI 3317 ...